```
/*
 * Copyright (c) 2009-2012, Salvatore Sanfilippo <antirez at gmail dot com>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *   * Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *   * Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *   * Neither the name of Redis nor the names of its contributors may be used
 *     to endorse or promote products derived from this software without
 *     specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

#ifndef __CONFIG_H
#define __CONFIG_H

#ifdef __APPLE__
#include <fcntl.h> // for fcntl(fd, F_FULLFSYNC)
#include <AvailabilityMacros.h>
#endif

#ifdef __linux__
#include <features.h>
#include <fcntl.h>
#endif

/* Define redis_fstat to fstat or fstat64() */
#if defined(__APPLE__) && !defined(MAC_OS_X_VERSION_10_6)
#define redis_fstat fstat64
#define redis_stat stat64
#else
#define redis_fstat fstat
#define redis_stat stat
#endif

/* Test for proc filesystem */
#ifdef __linux__
#define HAVE_PROC_STAT 1
#define HAVE_PROC_MAPS 1
#define HAVE_PROC_SMAPS 1
#define HAVE_PROC_SOMAXCONN 1
#define HAVE_PROC_OOM_SCORE_ADJ 1
#endif
```

```
/* Test for task_info() */
#if defined(__APPLE__)
#define HAVE_TASKINFO 1
#endif

/* Test for somaxconn check */
#if defined(__APPLE__) || defined(__FreeBSD__)
#define HAVE_SYSCTL_KIPC_SOMAXCONN 1
#elif defined(__OpenBSD__)
#define HAVE_SYSCTL_KERN_SOMAXCONN 1
#endif

/* Test for backtrace() */
#if defined(__APPLE__) || (defined(__linux__) && defined(__GLIBC__)) || \
    defined(__FreeBSD__) || ((defined(__OpenBSD__) || defined(__NetBSD__) ||
defined(__sun)) && defined(USE_BACKTRACE))\
 || defined(__DragonFly__) || (defined(__UCLIBC__) &&
defined(__UCLIBC_HAS_BACKTRACE__))
#define HAVE_BACKTRACE 1
#endif

/* MSG_NOSIGNAL. */
#ifdef __linux__
#define HAVE_MSG_NOSIGNAL 1
#if defined(SO_MARK)
#define HAVE_SOCKOPTMARKID 1
#define SOCKOPTMARKID SO_MARK
#endif
#endif

/* Test for polling API */
#ifdef __linux__
#define HAVE_EPOLL 1
#endif

/* Test for accept4() */
#ifdef __linux__
#define HAVE_ACCEPT4 1
#endif

#if (defined(__APPLE__) && defined(MAC_OS_X_VERSION_10_6)) || defined(__FreeBSD__)
|| defined(__OpenBSD__) || defined (__NetBSD__)
#define HAVE_KQUEUE 1
#endif

#ifdef __sun
#include <sys/feature_tests.h>
#ifdef _DTRACE_VERSION
#define HAVE_EVPORT 1
#define HAVE_PSINFO 1
#endif
#endif

/* Define redis_fsync to fdatasync() in Linux and fsync() for all the rest */
#if defined(__linux__)
#define redis_fsync(fd) fdatasync(fd)
#elif defined(__APPLE__)
#define redis_fsync(fd) fcntl(fd, F_FULLFSYNC)
#else
```

```
#define redis_fsync(fd) fsync(fd)
#endif

#if defined(__FreeBSD__)
#if defined(SO_USER_COOKIE)
#define HAVE_SOCKOPTMARKID 1
#define SOCKOPTMARKID SO_USER_COOKIE
#endif
#endif

#if defined(__OpenBSD__)
#if defined(SO_RTABLE)
#define HAVE_SOCKOPTMARKID 1
#define SOCKOPTMARKID SO_RTABLE
#endif
#endif

#if __GNUC__ >= 5 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 5)
#define redis_unreachable __builtin_unreachable
#else
#define redis_unreachable abort
#endif

#if __GNUC__ >= 3
#define likely(x) __builtin_expect(!!(x), 1)
#define unlikely(x) __builtin_expect(!!(x), 0)
#else
#define likely(x) (x)
#define unlikely(x) (x)
#endif

#if defined(__has_attribute)
#if __has_attribute(no_sanitize)
#define REDIS_NO_SANITIZE(sanitizer) __attribute__((no_sanitize(sanitizer)))
#endif
#endif
#if !defined(REDIS_NO_SANITIZE)
#define REDIS_NO_SANITIZE(sanitizer)
#endif

/* Define rdb_fsync_range to sync_file_range() on Linux, otherwise we use
 * the plain fsync() call. */
#if (defined(__linux__) && defined(SYNC_FILE_RANGE_WAIT_BEFORE))
#define HAVE_SYNC_FILE_RANGE 1
#define rdb_fsync_range(fd,off,size)
sync_file_range(fd,off,size,SYNC_FILE_RANGE_WAIT_BEFORE|SYNC_FILE_RANGE_WRITE)
#elif defined(__APPLE__)
#define rdb_fsync_range(fd,off,size) fcntl(fd, F_FULLFSYNC)
#else
#define rdb_fsync_range(fd,off,size) fsync(fd)
#endif

/* Check if we can use setproctitle().
 * BSD systems have support for it, we provide an implementation for
 * Linux and osx. */
#if (defined __NetBSD__ || defined __FreeBSD__ || defined __OpenBSD__)
#define USE_SETPROCTITLE
#endif

#if defined(__HAIKU__)
```

```
#define ESOCKTNOSUPPORT 0
#endif

#if (defined __linux || defined __APPLE__)
#define USE_SETPROCTITLE
#define INIT_SETPROCTITLE_REPLACEMENT
void spt_init(int argc, char *argv[]);
void setproctitle(const char *fmt, ...);
#endif

/* Byte ordering detection */
#include <sys/types.h> /* This will likely define BYTE_ORDER */

#ifndef BYTE_ORDER
#if (BSD >= 199103)
# include <machine/endian.h>
#else
#if defined(linux) || defined(__linux__)
# include <endian.h>
#else
#define LITTLE_ENDIAN   1234    /* least-significant byte first (vax, pc) */
#define BIG_ENDIAN      4321    /* most-significant byte first (IBM, net) */
#define PDP_ENDIAN      3412    /* LSB first in word, MSW first in long (pdp)*/

#if defined(__i386__) || defined(__x86_64__) || defined(__amd64__) || \
   defined(vax) || defined(ns32000) || defined(sun386) || \
   defined(MIPSEL) || defined(_MIPSEL) || defined(BIT_ZERO_ON_RIGHT) || \
   defined(__alpha__) || defined(__alpha)
#define BYTE_ORDER    LITTLE_ENDIAN
#endif

#if defined(sel) || defined(pyr) || defined(mc68000) || defined(sparc) || \
    defined(is68k) || defined(tahoe) || defined(ibm032) || defined(ibm370) || \
    defined(MIPSEB) || defined(_MIPSEB) || defined(_IBMR2) || defined(DGUX) ||\
    defined(apollo) || defined(__convex__) || defined(_CRAY) || \
    defined(__hppa) || defined(__hp9000) || \
    defined(__hp9000s300) || defined(__hp9000s700) || \
    defined (BIT_ZERO_ON_LEFT) || defined(m68k) || defined(__sparc)
#define BYTE_ORDER      BIG_ENDIAN
#endif
#endif /* linux */
#endif /* BSD */
#endif /* BYTE_ORDER */

/* Sometimes after including an OS-specific header that defines the
 * endianness we end with __BYTE_ORDER but not with BYTE_ORDER that is what
 * the Redis code uses. In this case let's define everything without the
 * underscores. */
#ifndef BYTE_ORDER
#ifdef __BYTE_ORDER
#if defined(__LITTLE_ENDIAN) && defined(__BIG_ENDIAN)
#ifndef LITTLE_ENDIAN
#define LITTLE_ENDIAN __LITTLE_ENDIAN
#endif
#ifndef BIG_ENDIAN
#define BIG_ENDIAN __BIG_ENDIAN
#endif
#if (__BYTE_ORDER == __LITTLE_ENDIAN)
#define BYTE_ORDER LITTLE_ENDIAN
#else
```

```
#define BYTE_ORDER BIG_ENDIAN
#endif
#endif
#endif
#endif

#if !defined(BYTE_ORDER) || \
    (BYTE_ORDER != BIG_ENDIAN && BYTE_ORDER != LITTLE_ENDIAN)
        /* you must determine what the correct bit order is for
         * your compiler - the next line is an intentional error
         * which will force your compiles to bomb until you fix
         * the above macros.
         */
#error "Undefined or invalid BYTE_ORDER"
#endif

#if (__i386 || __amd64 || __powerpc__) && __GNUC__
#define GNUC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)
#if defined(__clang__)
#define HAVE_ATOMIC
#endif
#if (defined(__GLIBC__) && defined(__GLIBC_PREREQ))
#if (GNUC_VERSION >= 40100 && __GLIBC_PREREQ(2, 6))
#define HAVE_ATOMIC
#endif
#endif
#endif

/* Make sure we can test for ARM just checking for __arm__, since sometimes
 * __arm is defined but __arm__ is not. */
#if defined(__arm) && !defined(__arm__)
#define __arm__
#endif
#if defined (__aarch64__) && !defined(__arm64__)
#define __arm64__
#endif

/* Make sure we can test for SPARC just checking for __sparc__. */
#if defined(__sparc) && !defined(__sparc__)
#define __sparc__
#endif

#if defined(__sparc__) || defined(__arm__)
#define USE_ALIGNED_ACCESS
#endif

/* Define for redis_set_thread_title */
#ifdef __linux__
#define redis_set_thread_title(name) pthread_setname_np(pthread_self(), name)
#else
#if (defined __FreeBSD__ || defined __OpenBSD__)
#include <pthread_np.h>
#define redis_set_thread_title(name) pthread_set_name_np(pthread_self(), name)
#elif defined __NetBSD__
#include <pthread.h>
#define redis_set_thread_title(name) pthread_setname_np(pthread_self(), "%s", name)
#elif defined __HAIKU__
#include <kernel/OS.h>
#define redis_set_thread_title(name) rename_thread(find_thread(0), name)
#else
```

```
#if (defined __APPLE__ && defined(MAC_OS_X_VERSION_10_7))
int pthread_setname_np(const char *name);
#include <pthread.h>
#define redis_set_thread_title(name) pthread_setname_np(name)
#else
#define redis_set_thread_title(name)
#endif
#endif
#endif

/* Check if we can use setcpuaffinity(). */
#if (defined __linux || defined __NetBSD__ || defined __FreeBSD__ || defined
__DragonFly__)
#define USE_SETCPUAFFINITY
void setcpuaffinity(const char *cpulist);
#endif

/* deprecate unsafe functions
 *
 * NOTE: We do not use the poison pragma since it
 * will error on stdlib definitions in files as well*/
#if (__GNUC__ && __GNUC__ >= 4) && !defined __APPLE__
int sprintf(char *str, const char *format, ...) __attribute__((deprecated("please
avoid use of unsafe C functions. prefer use of snprintf instead")));
char *strcpy(char *restrict dest, const char *src) __attribute__((deprecated("please
avoid use of unsafe C functions. prefer use of redis_strlcpy instead")));
char *strcat(char *restrict dest, const char *restrict src)
__attribute__((deprecated("please avoid use of unsafe C functions. prefer use of
redis_strlcat instead")));
#endif

/* Test for posix_fadvise() */
#if defined(__linux__) || __FreeBSD__ >= 10
#define HAVE_FADVISE
#endif

#endif
```